Nios® II

Nios II System Architect

Design Tutorial

101 Innovation Drive
San Jose, CA 95134
www.altera.com

TU-01004-1.0

Copyright © 2009 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

I.S. EN ISO 9001

Contents

Chapter 1. Getting Started

Tutorial Introduction . . . 1–1
Hardware and Software Requirements . . . 1–1
Acquire the NEEK . . . 1–1
Install the Design Software . . . 1–2
Install the USB Device Driver . . . 1–2
Extract the Tutorial Files . . . 1–2

Chapter 2. Designing the System

Introduction . . . 2–1
Design Flow . . . 2–2
Design Requirements . . . 2–3
Design Strategy . . . 2–4
Block Diagram . . . 2–6

Chapter 3. Building the SOPC System

Run the Quartus II Software and SOPC Builder . . . 3–1
Specify the SOPC Builder System Components . . . 3–1
Add a Nios II Processor to Your System . . . 3–3
Add a DDR SDRAM High Performance Controller . . . 3–4
Add a Clock-Crossing Bridge Between the CPU and DDR SDRAM Clock Domains . . . 3–5
Add a Scatter-Gather DMA Controller . . . 3–7
Coordinate Components in the System . . . 3–9
Specify External Clocks and Clock Connections . . . 3–9
Reassign Component Base Addresses to Eliminate Memory Conflicts . . . 3–11
Set Interrupt Priorities . . . 3–11
Set Arbitration Priorities . . . 3–12
Specify the Nios II Processor Boot Configuration . . . 3–13
Generate the System . . . 3–15

Chapter 4. Completing the Quartus II Project

Complete the Quartus II Project . . . 4–1
Open the Quartus II Project Block Diagram . . . 4–1
Add the SOPC Builder System to the Quartus II Project . . . 4–2
Add Timing Constraints and Compile Your Design . . . 4–3
Configure the FPGA . . . 4–4

Chapter 5. Creating the Software Application

Create a New Software Project in the Nios II IDE . . . 5–1
Add Source Code Files to the Project . . . 5–3
Configure System Library Build Properties . . . 5–3
Specify the Zip File System Settings . . . 5–4
Build the Software Project . . . 5–5
Program the Zip File to Flash Memory . . . 5–5
Run the Software Application on the NEEK . . . 5–7
Console Output . . . 5–7
Interact with the Software Application . . . 5–8

Additional Information

Taking the Next Step Info–1
How to Contact Altera Info–2
Revision History Info–2
Typographic Conventions Info–2

1. Getting Started

Tutorial Introduction

This tutorial teaches you how to use SOPC Builder, the Quartus® II software, and the Nios® II Embedded Design Suite (EDS) to implement a digital picture viewer in a Cyclone III FPGA. In this tutorial, you build a processor-based hardware system in programmable logic for the Nios II Embedded Evaluation Kit, Cyclone® III Edition (NEEK) and run software on it. The tutorial demonstrates the steps to build complete, complex embedded designs using the Altera® design software.

You must use the specified spelling for names of files, components, and other objects, for the tutorial design to function correctly with the software provided. For example, the software application refers to hardware components by the names specified in this tutorial. If you name a component differently, the software application does not identify it correctly.

Hardware and Software Requirements

This tutorial requires the following hardware and software:

- NEEK development board
- Quartus II software v9.0—FPGA synthesis and compilation tool that contains SOPC Builder and the MegaCore® IP library, including the Nios II embedded processor
- Nios II EDS 9.0—Complete environment for Nios II software development that includes the Nios II Integrated Development Environment (IDE)
- The design files for this tutorial

This section describes the following procedures:

1. Acquire the NEEK
2. Install the Design Software
3. Install the USB Device Driver
4. Extract the Tutorial Files

After you complete all these setup and installation requirements, you are ready to create your first system-level design.

Acquire the NEEK

You can purchase the NEEK by following the instructions on the Nios II Embedded Evaluation Kit, Cyclone III Edition web page.

Install the Design Software

This section teaches you how to install the Quartus II software v9.0 and the Nios II EDS v9.0. The design software is available on the Altera Complete Design Suite DVD or by downloading from the web.

Two installation options are available for the Altera Complete Design Suite, a subscription package and a free package.

The subscription package requires a paid subscription license, which entitles you to a full license for Altera's MegaCore IP library base suite. The free package requires you to register but does not require that you pay for a license. It includes the free Quartus II Web Edition version of the tools. If you are using the free package, follow the prompts to obtain your Web Edition license file. This license enables the Quartus II Web Edition software to generate the FPGA programming files required for this tutorial.

To install the Altera Complete Design Suite, perform one of the following steps:

- Insert the Altera Complete Design Suite DVD-ROM and follow the installation prompts.
- Download the Subscription Edition or the Web Edition of the Quartus II software from https://www.altera.com/support/software/download/sof-download_center.html.

For information about installing the Quartus II software, refer to *Quartus II Installation & Licensing for Windows and Linux Workstations*.

Install the USB Device Driver

After the software is installed, you can connect the NEEK board to the USB port on your PC. If the software prompts you to install the USB Blaster driver, install it from the following location:
*<Altera software installation directory>***\90\quartus\drivers\usb-blaster**. If the driver is not located in this directory, install the appropriate version for your system from the following web page: www.altera.com/support/software/drivers.

For additional information, refer to the *USB-Blaster Download Cable User Guide*.

Extract the Tutorial Files

The tutorial files are available for download as a compressed zip file (**.zip**) adjacent to this tutorial on the Altera literature web pages. When you extract the files from this **.zip**, you must ensure that the directory path on your PC contains only folder names with no spaces in them. The directory in which you unzip the tutorial files is your working directory.

2. Designing the System

Introduction

To architect a system, you specify the design requirements and develop a suitable design strategy to address each requirement. Design requirements include immediate design goals of the product and features to keep your product commercially successful in the future. You must consider desired application functionality, cost, flexibility to add future features, time to market, and obsolescence.

Programmable logic solutions provide optimal flexibility in your architecture to add features over time quickly and at low cost. For example, a programmable logic solution enables you to add features such as graphics or video display to a commercial implementation of the digital picture viewer quickly and cheaply. Programmable logic solutions allow you to integrate all the parts of the design on a single low-cost, low-power programmable logic device. They support short time to market, and insulate your design from part obsolescence — the potential lack of future support for specific microprocessors, LCD panels, LCD controllers, and flash memory devices, for example — by providing the ability to migrate your design to another device without extensive modification.

Therefore, this tutorial implements the digital picture viewer on a programmable logic platform using Altera's SOPC Builder.

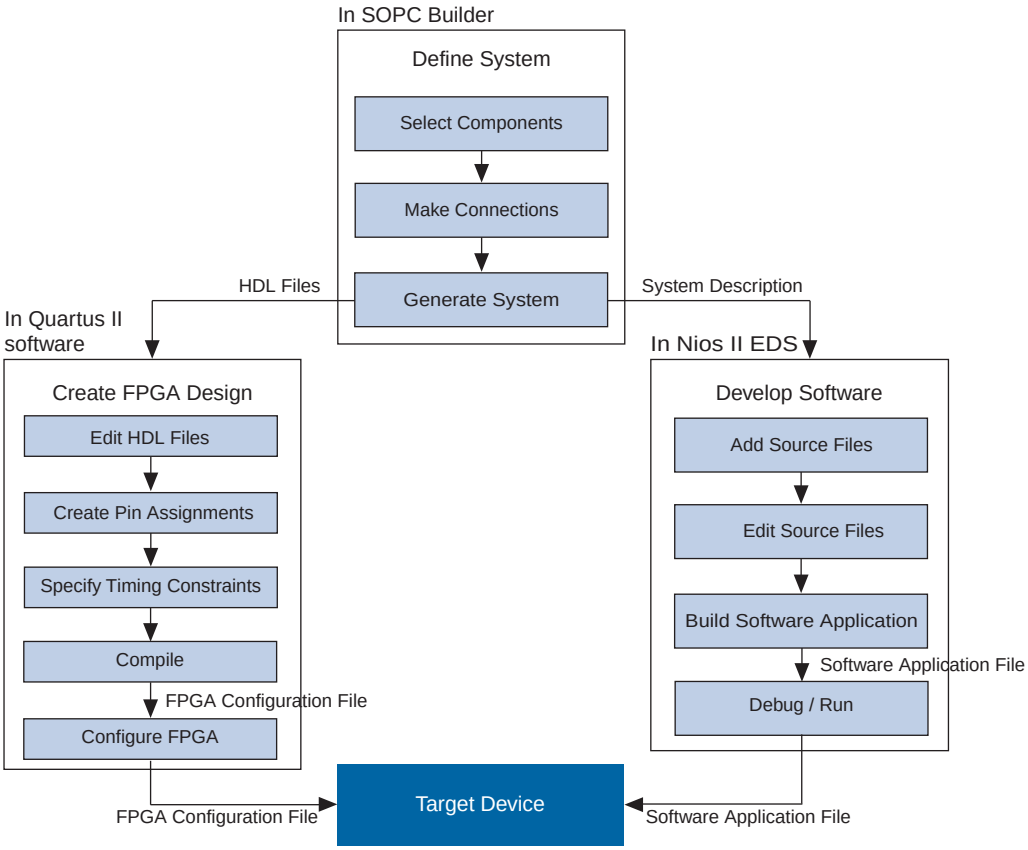
Product design goals originate in customer requirements and become inputs to system definition. System definition is the first step in the design process.

The following sections describe the Altera design flow for an embedded system, the design requirements for the digital picture viewer design you implement in this tutorial, the design strategies that dictate the decisions about the hardware components in the implemented design, and the basic system description for the digital picture viewer. By explaining the design flow, design requirements, and design strategy in this chapter, this chapter teaches you the system architecture of the digital picture viewer.

Design Flow

Figure 2–1 depicts the typical flow for embedded system design.

Figure 2–1. System Design Flow Using SOPC Builder



In this flow, you specify the system definition using SOPC Builder. After you define the SOPC Builder system, SOPC Builder generates the following two kinds of output:

- The HDL files that the Quartus II software compiles to generate the configuration file for the FPGA. This Quartus II compilation process is the hardware flow.
- A system description that the software development tools use to generate a system library specific to the SOPC Builder system. This system library, also called a board support package, supports the Nios II processor in running the software. The Nios II IDE provides an environment in which you can develop software applications for your system. This Nios II IDE development process is the software flow.

The output of the hardware flow is an FPGA image that is used to configure the target device. The output of the software flow is an executable file that the Nios II processor can run.

In this tutorial you perform the following steps:

1. Build an SOPC system:
   a. Open the Quartus II project.
   b. Start SOPC Builder
   c. Select and configure system components, including IP MegaCore functions
   d. Make connections
   e. Assign clocks and set base addresses
   f. Set interrupt and arbitration priorities
2. Generate the system to create the following items:
   - HDL for the entire SOPC Builder system
   - A system description file that software development tools use to build the hardware drivers and other relevant system information for the software application
3. Complete the Quartus II project:
   a. Generate a schematic diagram for the SOPC system
   b. Add the SOPC Builder system to the top level of the Quartus II project
   c. Add timing constraints
   d. Compile the project to generate an FPGA programming file (**.sof**)
   e. Use the Quartus II Programmer to download the **.sof** to the FPGA
4. Develop the software application:
   a. Start the Nios II IDE
   b. Add source files
   c. Configure build properties
   d. Build the application to generate the executable file (**.elf**) and the flash memory image file (**.flash**)
5. Use the Flash Programmer to download the flash image data to CFI flash memory
6. Use the Nios II IDE to download and run the **.elf** for the software application

Each major step of the design process is described in a tutorial chapter.

Design Requirements

In this tutorial, you design a digital picture viewer that allows an user to view pictures stored in JPEG format and to scroll through JPEG pictures stored in the external flash memory, using the LCD touch-screen display.

Design Strategy

SOPC Builder provides a menu of standard hardware components that you can add to your system. You can add several instances of each component. The only restriction is the resources available on your board to implement all of the components you instantiate in your design.

The NEEK comprises a Cyclone III FPGA Starter Board and an LCD Multimedia High Speed Mezzanine Card (HSMC) daughtercard in addition to accessories and software. The following NEEK hardware resources are required to run the design you create in this tutorial:

- The following Cyclone III FPGA Starter Board resources are required:
  - Cyclone III EP3C25F324 FPGA
  - Embedded USB-Blaster™ circuitry (including an Altera EPM3128A CPLD) allowing download of FPGA configuration files through your PC's USB port
  - 256-Mbit DDR SDRAM
  - 1 MByte of synchronous SRAM
  - 16 MBytes of Intel P30/P33 flash memory
  - 50-MHz on-board oscillator
- The following LCD daughtercard resources are required:
  - LCD touch-screen 800 × 480 pixel display
  - 10-bit VGA digital access card

The NEEK has additional features, including additional memory, not required to implement this design.

The SOPC Builder system contains both control path and data path components. The Nios II processor is the main system controller. It initializes and calibrates the LCD controller, intercepts user touch input, and initializes the data path. It communicates with other control path components through Avalon Memory-Mapped (Avalon-MM) interface ports.The Nios II processor reads the program code from low-latency SSRAM memory, and the DDR SDRAM holds the video frame buffer. A tristate bridge component enables pin sharing between the SSRAM memory and the flash memory on the NEEK.

The Nios II processor sends frame buffer data from the DDR SDRAM to a video pipeline of specialized Avalon Streaming (Avalon-ST) components that move and process the pixel data, converting it to video data signals and sending them to the LCD screen.

The pixels must be rendered on the LCD screen smoothly and without delays. This design achieves this goal by running the DDR SDRAM at a high clock rate to improve the performance of the video frame buffer. The DDR SSRAM memory controller component runs at half rate, and a clock-crossing bridge enables smooth data transfer between the DDR memory controller and CPU clock domains.

Table 2–1 shows how the individual design requirements for this project are implemented by specific design strategies, which in turn dictate the components you add to your system in this tutorial.

Table 2–1. Standard Hardware Components to Implement the Digital Picture Viewer Requirements

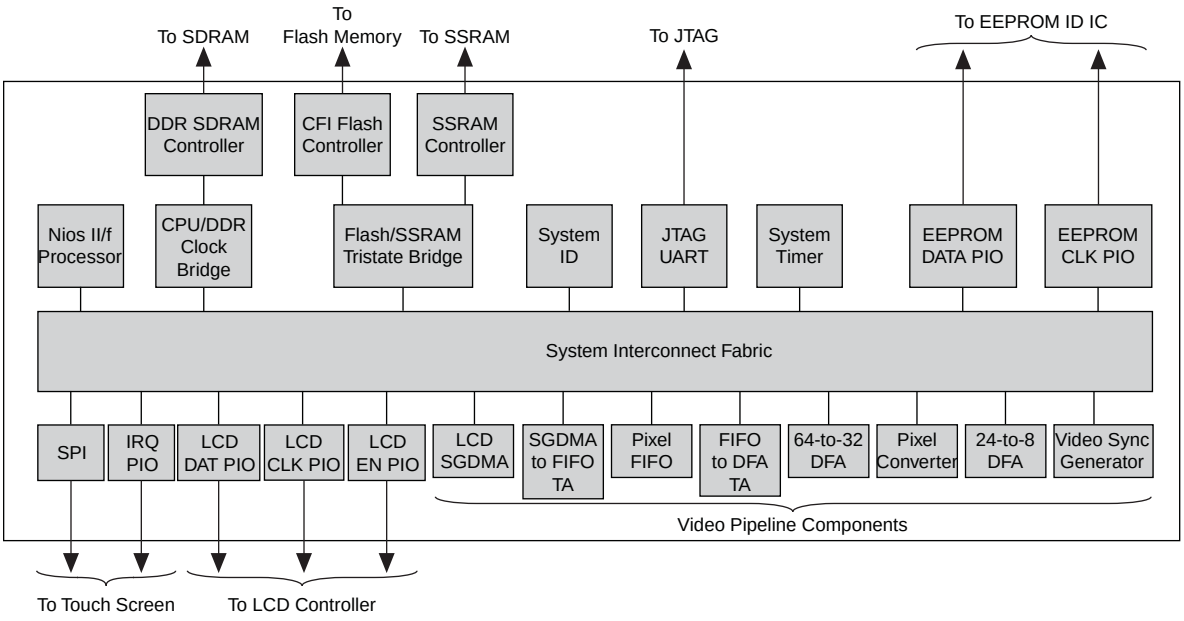
Design Requirement	Design Strategy	Components Needed
Display JPEG pictures on the LCD Display	1. LCD screen controller 2. A video pipeline to send video data to the LCD screen	1. PIO 2. Video pipeline peripherals
Scroll through JPEG pictures	1. An interrupt to capture touch screen events 2. Touch screen control	1. PIO 2. SPI
Frame buffering	1. Need a large memory. Use the on-board DDR SDRAM 2. Transfer video data from memory to Video pipeline	1. DDR SDRAM Controller 2. SGDMA
Store JPEG images, application software, and reset vectors in a non-volatile memory	Use the on-board flash memory	Flash Controller
Store program code, stack, and exception vectors	Use the on-board SSRAM	SSRAM Controller
Generate interrupts and measure system performance	Need a timer	Interval Timer
DOwnload, run, and debug software	JTAG UART communication interfaces	JTAG UART
Keep track of hardware and software builds	Ensure software is only run on the hardware for which it was compiled	System ID peripheral

The implementation of the video pipeline and the touch screen controller are described in more detail in *AN527: Implementing an LCD Controller* and in *Nios II 3C25 Microprocessor with LCD Controller Data Sheet*.

Block Diagram

Figure 2–2 shows a high-level block diagram of the system you develop to implement the design.

Figure 2–2. High-Level Block Diagram of Digital Picture Viewer



As systems get larger and more complicated, it is easier to design at a higher level of abstraction using—and reusing—IP MegaCore functions and standard components. SOPC Builder automatically generates the system interconnect fabric, which is the glue logic required to connect the design blocks together. The system interconnect fabric manages design issues such as dynamic bus-width matching, interrupt priorities, and arbitration.

For additional information about the Altera system interconnect fabric, refer to *Avalon Interface Specifications*.

For additional information about SOPC Builder, refer to the Altera SOPC Builder literature page.

3. Building the SOPC System

In this chapter you add standard and custom components to the system, make connections where required, assign the clocks, set arbitration priorities, and generate the system.

The tutorial SOPC Builder system contains 24 components. You start with a partially completed SOPC Builder system, to which you add a Nios II embedded processor, a DDR SDRAM high-performance controller, an Avalon-MM clock-crossing bridge peripheral, and a scatter-gather DMA controller. A table describes the function of each of the components.

After you add the components to the system, you must perform some additional steps to make the components work together. All of the necessary steps are included in this chapter.

If you wish to skip the steps to create the FPGA configuration file (**.sof**), you can skip this chapter and use the Quartus II Archive file (**.qar**) available in your working directory.

Run the Quartus II Software and SOPC Builder

To run SOPC Builder, perform the following steps:

1. To run the Quartus II software, on your Windows system, click **Start > All Programs > Altera > Quartus II** *<version>*.
2. On the File menu, click **Open Project**.
3. Browse to your working directory and highlight **neek_hw_lab.qpf**.
4. Click **Open**.

   The Quartus II project for the tutorial opens. It is missing the Nios II processor and some of the other components that implement the digital picture viewer. In the following steps, you add these components.
5. On the Tools menu, click **SOPC Builder**.

   SOPC Builder opens with a system that is missing a few key components.

In the following section, you add the missing components to the system.

Specify the SOPC Builder System Components

The partially completed SOPC Builder system provided in your Quartus II project contains the LCD controller, video pipeline, and various other peripherals. This section teaches you how to specify the missing components to add to your SOPC Builder system. You add four critical components to your SOPC Builder system.

Table 3–1 shows the components in the partial system provided in your Quartus II project.

Table 3–1. Components in the Partial SOPC Builder System

Component	Component Instance Name	Component Role
System ID	sysid	Allows the Nios II IDE to verify that the software is built for the correct hardware version
Avalon-MM Tristate Bridge	flash_ssram_tristate_bridge	Allows pin-sharing to better enable the FPGA to drive multiple external devices, such as SRAM and flash memory
Cypress CY7C1380, under SRAM	ssram	This SRAM controller controls the timing for driving read and write transactions on this external SRAM. The SRAM stores the software program code, stack, and exception sections.
Flash Memory (CFI)	ext_flash	This Flash memory controller controls the timing for driving read and write transactions on the external CFI flash memory. The CFI flash memory device stores Nios II application software, the CPU reset vector, and the JPEG images for the picture viewer application.
JTAG UART	jtag_uart	Enables software to access a debug serial port
Interval timer	sys_clk_timer	Enables software to perform periodic interrupts for maintenance and to maintain software application timing requirements.
PIO (Parallel I/O)	lcd_i2c_scl	LCD controller interface components. The three-wire interface includes a data signal, a clock signal, and an enable signal. Refer to *AN527: Implementing an LCD Controller*.
PIO (Parallel I/O)	lcd_i2c_en	
PIO (Parallel I/O)	lcd_i2c_sdat	
PIO (Parallel I/O)	pio_id_eeprom_scl	Two-wire EEPROM ID interface components. The I2C serial EEPROM ID chip stores information about the board, including the touch screen calibration data.
PIO (Parallel I/O)	pio_id_eeprom_dat	
SPI (3 Wire Serial)	touch_panel_spi	Touch screen interface components. Refer to *AN527: Implementing an LCD Controller*.
PIO (Parallel I/O)	touch_panel_pen_irq_n	
Avalon-ST Timing Adapter	lcd_ta_sgdma_to_fifo	Video pipeline components.Refer to *AN527: Implementing an LCD Controller*.
On-Chip Memory	lcd_pixel_fifo	
Avalon-ST Timing Adapter	lcd_ta_fifo_to_dfa	
Avalon-ST Data Format Adapter	lcd_64_to_32_bits_dfa	
Pixel Converter (BGR0 -> BGR)	lcd_pixel-converter	
Avalon-ST Data Format Adapter	lcd_24_to_8_bits_dfa	
Video Sync Generator	lcd_sync_generator	

Ignore the error messages that appear in the SOPC Builder message console. When you perform the steps in "Coordinate Components in the System" on page 3–9, these errors are resolved.

This section contains the instructions that show you how to perform the following actions:

1. Add a Nios II Processor to Your System

2. Add a DDR SDRAM High Performance Controller
3. Add a Clock-Crossing Bridge Between the CPU and DDR SDRAM Clock Domains
4. Add a Scatter-Gather DMA Controller

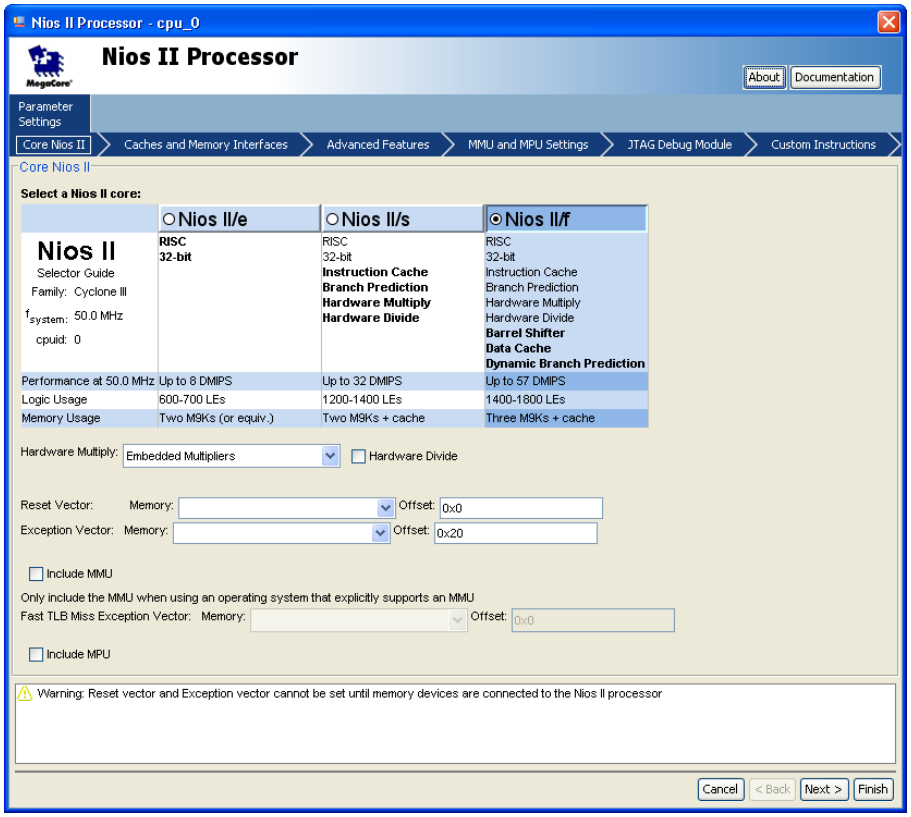
Altera recommends that you save your work after you add each component. To save the current SOPC Builder system, on the File menu, click **Save**.

Add a Nios II Processor to Your System

Your system requires a CPU to run the software application that controls the tutorial demo. To add a Nios II embedded processor to your system, perform the following steps:

1. Under **System Contents**, double-click **Nios II Processor**.
2. Select the Nios II/f processor. Figure 3–1 shows the Nios II Processor MegaWizard interface after you select the Nios II/f processor.
3. Click **Finish**.

Figure 3–1. Nios II Processor Component Settings



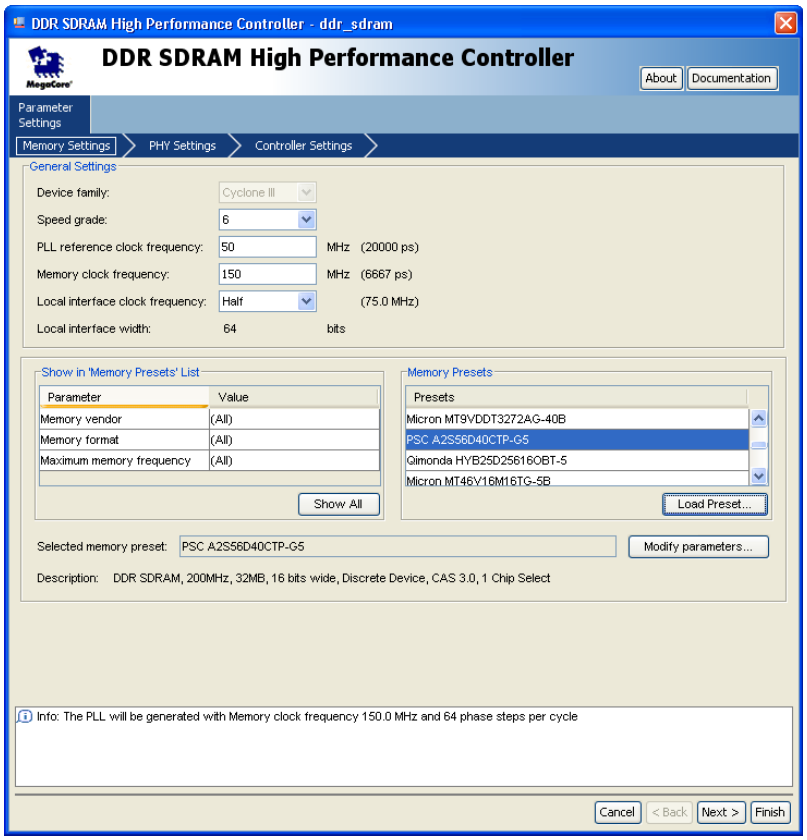
4. In the **Module Name** column, right-click the new Nios II processor name, click **Rename,** and rename the new component `cpu`.

Add a DDR SDRAM High Performance Controller

Your system requires a high-performance memory block to act as a video frame buffer. To add a high-performance DDR SDRAM controller to your system, perform the following steps:

1. Under **System Contents**, expand **Memories and Memory Controllers**, expand **SDRAM**, and double-click **DDR SDRAM High Performance Controller**.
2. On the **Memory Settings** tab, for **Speed grade**, select the correct speed grade for your device. This tutorial assumes speed grade 6.
3. For **PLL reference clock frequency**, type `50 MHz`.
4. For **Memory clock frequency**, type `150 MHz`.
5. For **Local interface clock frequency**, select **Half**.
6. Under **Memory Presets**, scroll and click **PSC A2S56D40CTP-G5** to match the DDR SDRAM on the NEEK. Figure 3–2 shows the **Memory Settings** tab after you assign these settings.

Figure 3–2. Memory Settings for DDR SDRAM High Performance Controller

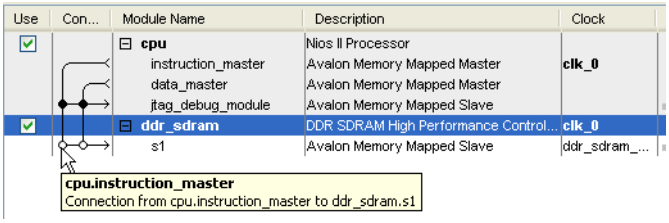


7. To keep the default settings on the **PHY Settings** and **Controller Settings** tabs, click **Finish.**
8. In the **Module Name** column, right-click the new DDR SDRAM High Performance Controller name, click **Rename,** and rename the new component `ddr_sdram`.

9. Now you must remove two connections between the cpu and ddr_sdram components that are created by default. The `cpu/instruction_master` port and `cpu/data_master` port should not be connected to the `ddr_sdram/s1` slave port, because the Nios II processor and the DDR SDRAM are in different clock regions.

   To remove these connections, in the **Connections** column, click on the filled at the intersection of each pair of relevant signal lines. After you click on a filled dot that represents a connection, an open dot indicates the two signals are no longer connected. Figure 3–3 shows the two severed connections, one of which is labeled.

Figure 3–3. Severed Connections Between the ddr_sdram s1 port and the CPU instruction_master and data_master Ports

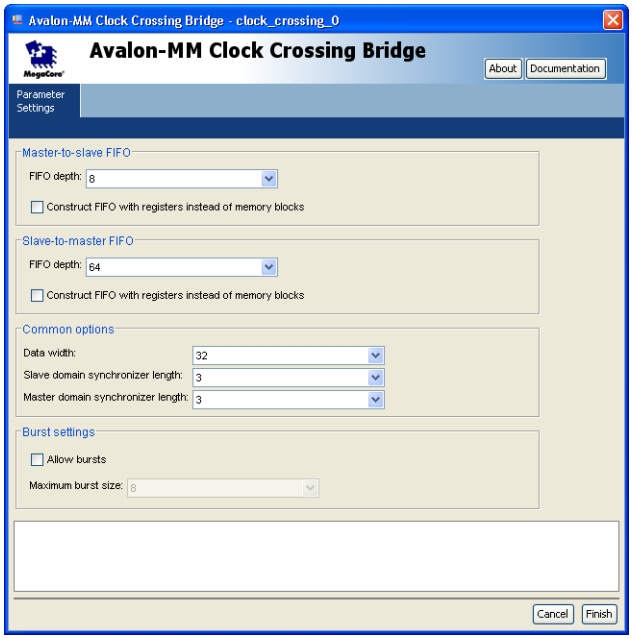


Add a Clock-Crossing Bridge Between the CPU and DDR SDRAM Clock Domains

To enhance the performance of the video frame buffer, the DDR SDRAM memory runs on a fast clock. Therefore, the Nios II processor and the DDR SDRAM memory run on different clocks. For optimal performance, your system requires a clock-domain crossing bridge between these two components. To add an Avalon-MM clock-crossing bridge component for DDR SDRAM to your system, perform the following steps:

1. Under **System Contents**, expand **Bridges and Adapters**, expand **Memory Mapped**, and double-click **Avalon-MM Clock Crossing Bridge**.
2. Under **Slave-to-master FIFO**, for **FIFO depth**, select **64**. Figure 3–4 shows the Avalon-MM Clock Crossing Bridge MegaWizard interface after you perform this step.

Figure 3–4. Avalon-MM Clock Crossing Bridge Component Settings



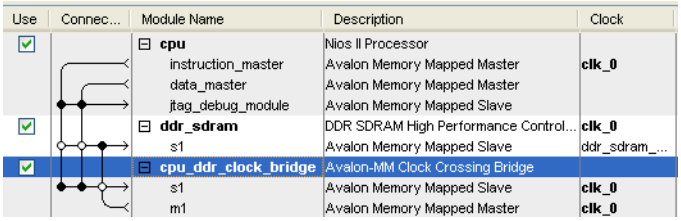
3. Click **Finish**.
4. In the **Module Name** column, right-click the Avalon-MM Clock Crossing Bridge component's name, click **Rename,** and rename the new component `cpu_ddr_clock_bridge`.
5. In the **Connections** column, ensure that the `cpu_ddr_clock-bridge/m1` master port is connected to the `ddr_sdram/s1` slave port.

   By default, the `cpu/instruction_master` and `cpu/data_master` ports are connected to the `cpu_ddr_clock_bridge/s1` slave port.

   If one of these connections is absent, click on the open dot at the intersection of the relevant signal lines to make the connection. A filled dot at the intersection indicates a connection is present. Move your mouse to the **Connections** column to make the connection dots visible.

   Figure 3–5 shows the desired cpu_ddr_clock_bridge port connections.

Figure 3–5. cpu_ddr_clock_bridge Component Connections



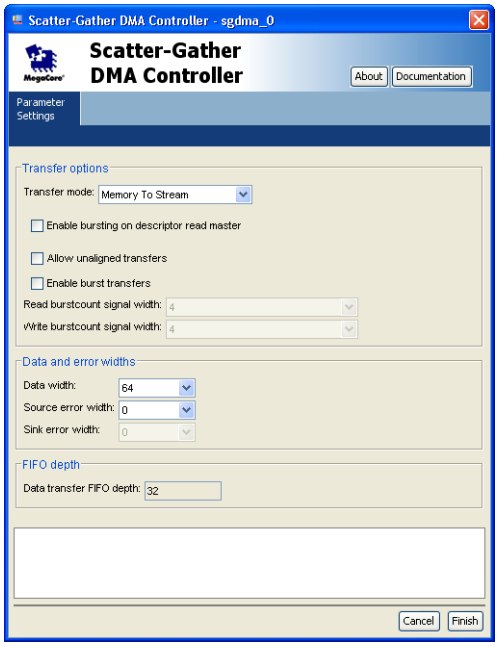
Add a Scatter-Gather DMA Controller

The scatter-gather DMA controller fetches data from the video frame buffer on the DDR SDRAM and sends the data to the video pipeline. To add a Scatter-Gather DMA Controller to your system, perform the following steps:

1. Under **System Contents**, expand **Memories and Memory Controllers**, expand **DMA**, and double-click **Scatter-Gather DMA Controller**.
2. Under **Transfer options**, for **Transfer mode**, select **Memory To Stream**.
3. Under **Data and error widths**, for **Data width**, select **64**.

   Figure 3–6 shows the Scatter-Gather DMA Controller MegaWizard interface after you perform this step.

Figure 3–6. Scatter-Gather DMA Controller Component Settings



4. Click **Finish**.
5. In the **Module Name** column, right-click the Scatter-Gather DMA Controller component's name, click **Rename,** and rename the new component `lcd_sgdma`.
6. In the **Connections** column, ensure that your system has the connections listed in Table 3–2.

Table 3–2. SGDMA Component Connections

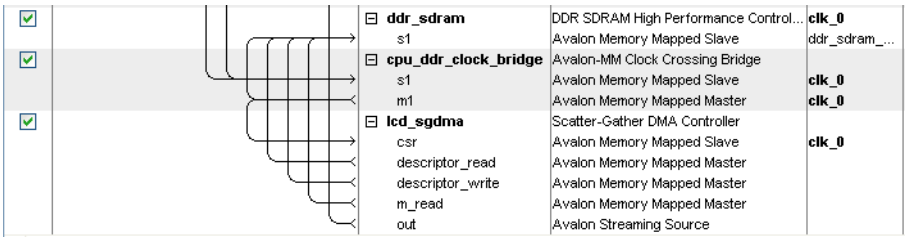
From	To	Description
`lcd_sgdma/csr`	`cpu_ddr_clock_bridge/m1`	Because the Nios II processor and the SGDMA component are in different clock domains, the `lcd_sgdma/csr` port must connect to the clock-crossing bridge. In "Coordinate Components in the System", you set the SGDMA component to use the same clock as the DDR SDRAM.
`lcd_sgdma/descriptor_read`	`ddr_sdram/s1`	The SGDMA fetches the read descriptors from the DDR SDRAM.
`lcd_sgdma/descriptor_write`	`ddr_sdram/s1`	The SGDMA fetches the write descriptors from the DDR SDRAM.
`lcd_sgdma/m_read`	`ddr_sdram/s1`	The SGDMA fetches the video frame buffer data from the DDR SDRAM.
`lcd_sgdma/out`	`lcd_ta_sgdma_to_fifo/in`	The SGDMA sends the data to the video pipeline.

7. In the **Connections** column, ensure that your system does not have the following connection:
   - `lcd_sgdma/csr` to `cpu/data_master` — Because the Nios II processor and the SGDMA component are in different clock domains, the `lcd_sgdma/csr` port must connect to the clock-crossing bridge and not directly to the Nios II processor.

Click on the dot at the intersection of the relevant signal lines to toggle the connection. A filled dot at the intersection indicates a connection is present, and an open dot indicates a connection is severed. Move your mouse to the **Connections** column to make the connection dots visible.

Figure 3–5 shows the desired `lcd_sgdma` port connections. The `lcd_sgdma/out` port to `lcd_ta_sgdma_to_fifo/in` port connection does not appear in the figure.

Figure 3–7. lcd_sgdma Component Connections



After you perform these instructions, your SOPC Builder system contains all of the components it requires to implement your design requirements. Now you must resolve the system validation errors.

Coordinate Components in the System

This section teaches you how to coordinate the components in your SOPC Builder system to ensure they work together correctly. It contains the instructions that show you how to perform the following actions:

1. Specify External Clocks and Clock Connections
2. Reassign Component Base Addresses to Eliminate Memory Conflicts
3. Set Interrupt Priorities
4. Set Arbitration Priorities
5. Specify the Nios II Processor Boot Configuration

Specify External Clocks and Clock Connections

You must create one additional clock in the SOPC Builder system, and connect each component to the correct clock. One of the output ports of the Quartus II project PLL connects to the new clock.

The clock source for the full digital picture viewer is the 50-MHz oscillator on the NEEK board. A PLL in the project that is outside your SOPC Builder system takes the oscillator clock as input and generates two 100-MHz clocks, a system clock, `cpu_clk`, and an external SSRAM clock, `ssram_clk`. The `ssram_clk` clocks the SSRAM memory itself and is not part of the SOPC Builder system. Because the PLL is outside the SOPC Builder system, clocks that the PLL generates are considered external clocks to the SOPC Builder system.

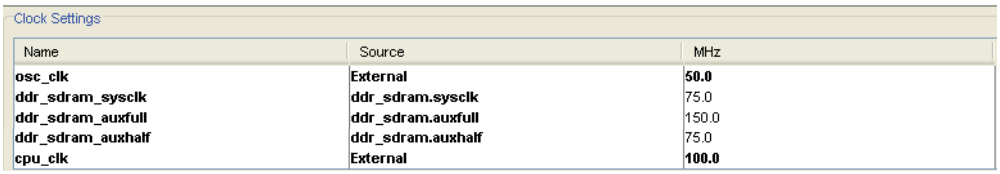
The 50-MHz oscillator is also the reference clock frequency for the DDR SDRAM controller, which runs at 150 MHz. The SDRAM runs its local interface at half-rate frequency, providing a 75-MHz clock that clocks some of the video pipeline components, such as the SGDMA component.

Create the new clock, set its frequency, and create clock connections by performing the following steps:

1. To rename the **clk_0** clock to `osc_clk`, in the **Clock Settings** box, in the **Name** column, double-click **clk_0**.
2. Type `osc_clk` to replace the current name **clk_0**. The name change propagates to the **Clock** column entries for the individual components automatically.
3. To add the new `cpu_clk` clock, in the **Clock Settings** box, to the right of the clocks list, click **Add**. A new clock appears at the bottom of the list, with the name **clk_0**.
4. To rename the new **clk_0** clock to `cpu_clk`, in the **Clock Settings** box, in the **Name** column, double-click **clk_0**.
5. Type `cpu_clk` to replace the current name **clk_0**.
6. In the **MHz** column, double-click in the **cpu_clk** row.
7. Type `100.0` to replace the default frequency of 50.0 MHz.
8. Ensure the value in the **Source** column is **External**.

   Figure 3–8 shows the SOPC Builder clocks list after you perform these steps:

Figure 3–8. Clocks for Digital Picture Viewer SOPC Builder System



9. If the clock ports for the components do not appear in the **Module Name** column, perform the following steps:
   a. Click the **Filters** button below your system display.
   b. In the **Filters** dialog box, under **Filter**, select **All**.
   c. Close the **Filters** dialog box.
10. To change the clocks for the individual components, for each component, click the **Clock** column and select the desired clock name according to Table 3–3.

    Table 3–3 lists the clocks for the different component ports. For each clock, components are listed in the order they appear in the SOPC Builder **System Contents** tab.

Table 3–3. Clocks for Individual Components in Digital Picture Viewer SOPC Builder System

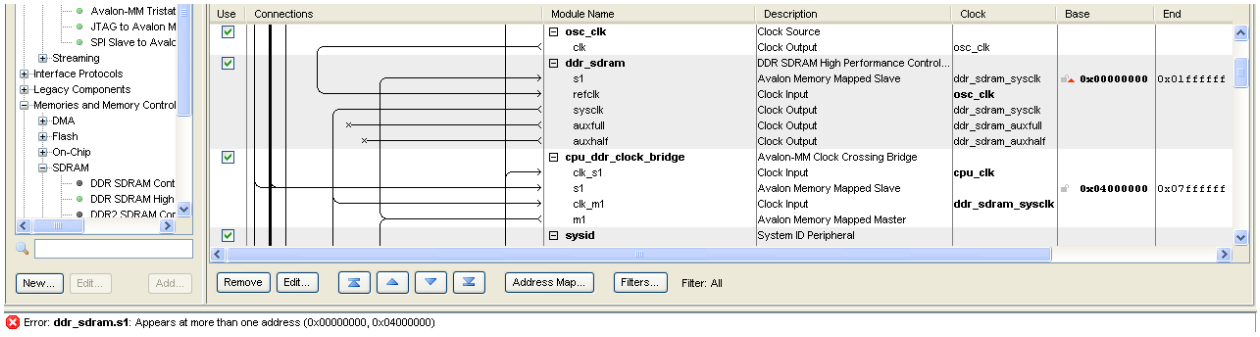
Clock Name	Frequency	Source	Component Port
osc_clk	50 MHz	Oscillator	ddr_sdram/refclk
ddr_sdram_sysclk	75 MHz	SDRAM PLL	lcd_ta_sgdma_to_fifo/clk lcd_pixel_fifo/clk_in ddr_sdram/s1 cpu_ddr_clock_bridge/clk_m1 lcd_sgdma/clk
cpu_clk	100 MHz	External PLL	sysid/clk flash_ssram_tristate_bridge/clk ssram/clk ext_flash/clk jtag_uart/clk sys_clk_timer/clk lcd_i2c_scl/clk lcd_i2c_en/clk lcd_i2c_sdat/clk pio_id_eeprom_scl/clk pio_id_eeprom_dat/clk touch_panel_spi/clk touch_panel_pen_irq_n/clk lcd_pixel_fifo/clk_out lcd_ta_fifo_to_dfa/clk lcd_64_to_32_bits_dfa/clk lcd_pixel_converter/clk lcd_24_to_8_bits_dfa/clk lcd_sync_generator/clk cpu/clk cpu_ddr_clock_bridge/clk_s1

Reassign Component Base Addresses to Eliminate Memory Conflicts

To reassign component base addresses to eliminate undesired overlap between the address spaces of different components, perform the following steps:

1. On the System menu, click **Auto-Assign Base Addresses**. The tool assigns appropriate base addresses for the components.
2. To remove the error shown in Figure 3–9, which the tool could not resolve automatically, reassign the `cpu_ddr_clock_bridge` base address manually in the **Base** column by double-clicking the address and typing `0x00000000`.

Figure 3–9. Base Address Setting Conflict that Requires Manual Reassignment



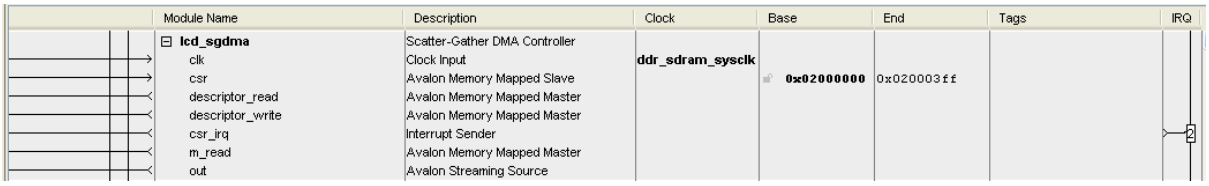
3. To view the address map for the system, click **Address Map**. The **Address Map** dialog box appears.
4. Click **Close**.

Set Interrupt Priorities

To render images on the LCD screen smoothly, the `lcd_sgdma` component must continuously service the frame buffer without stalling. When the `lcd_sgdma` completes a transaction, it must be updated immediately by the Nios II processor. Therefore, `lcd_sgdma` must have the highest interrupt priority, followed by the timer, jtag_uart, and touch-screen components. The lowest interrupt value indicates the highest interrupt priority. Assign interrupt priorities to the system components by performing the following steps for each interrupt-request (IRQ) port listed in Table 3–4:

1. In the **IRQ** column, double-click the box with the current interrupt priority setting. In Figure 3–10, the **IRQ** column is the rightmost column.

Figure 3–10. Interrupt Request Priority of lcd_sgdma Component



2. Type the interrupt priority value from Table 3–4. Figure 3–10 shows the `lcd_sgdma` component after you perform this step.

Table 3–4. Interrupt Priority Assignments

Component/Port	Interrupt Priority Value
`lcd_sgdma/csr_irq`	2
`sys_clk_timer/irq`	8
`jtag_uart/irq`	10
`touch_panel_pen_irq_n/irq`	14
`touch_panel_spi/irq`	16

Set Arbitration Priorities

To ensure that the video pipeline operates smoothly, you must assign the highest arbitration priority to the `lcd_sgdma` component to access the `ddr_sdram` component. In addition, because the heap memory is located in DDR SDRAM, you must increase the arbitration priority of the Nios II processor for `ddr_sdram` to support more back-to-back data transactions.

To assign arbitration priorities among the components in your system, perform the following steps:

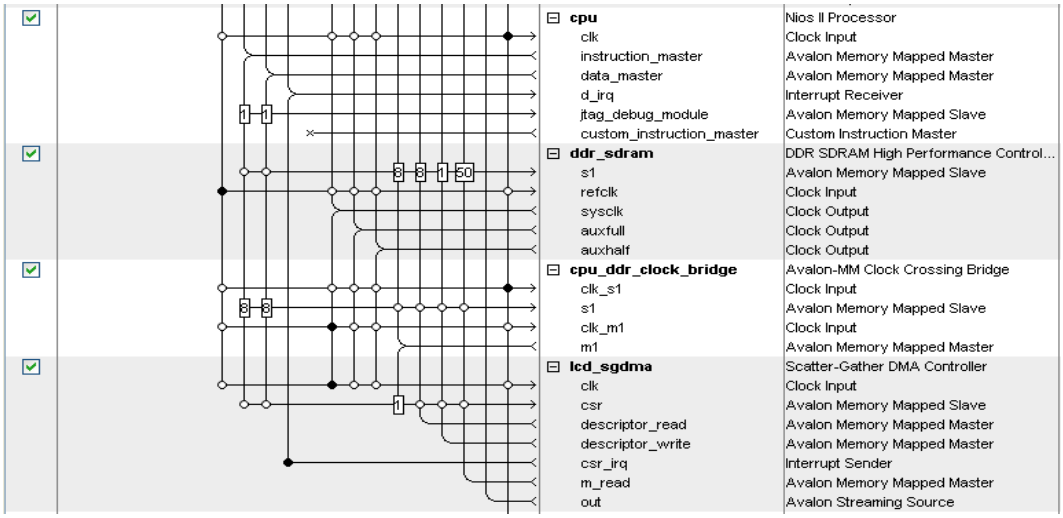
1. On the View menu, click **Show Arbitration**. The connection panel displays the arbitration priority of each master for each slave to which it is connected. By default, SOPC Builder assigns arbitration priority 1 for each connected master-slave pair.
2. For each master-slave connection in Table 3–5, double-click the box that represents the connection and type the new value from the table.

Table 3–5. Arbitration Priority Assignments

Master Port	Slave Port	Arbitration Priority
`lcd_sgdma/m_read`	`ddr_sdram/s1`	50
`lcd_sgdma/descriptor_read`	`ddr_sdram/s1`	8
`cpu_ddr_clock_bridge/m1`	`ddr_sdram/s1`	8
`cpu/instruction_master`	`cpu_ddr_clock_bridge/s1`	8
`cpu/data_master`	`cpu_ddr_clock-bridge/s1`	8

Figure 3–11 shows the arbitration values after you make these assignments.

Figure 3–11. Arbitration Priorities in Digital Picture Viewer SOPC Builder System



3. To hide the arbitration priorities, on the View menu, click **Show Arbitration**.

Specify the Nios II Processor Boot Configuration

Immediately after it completes its hardware reset sequence, the Nios II processor begins executing software located at a predefined memory location. The reset vector specifies this location.

Altera recommends that your reset vector specify a non-volatile memory location.

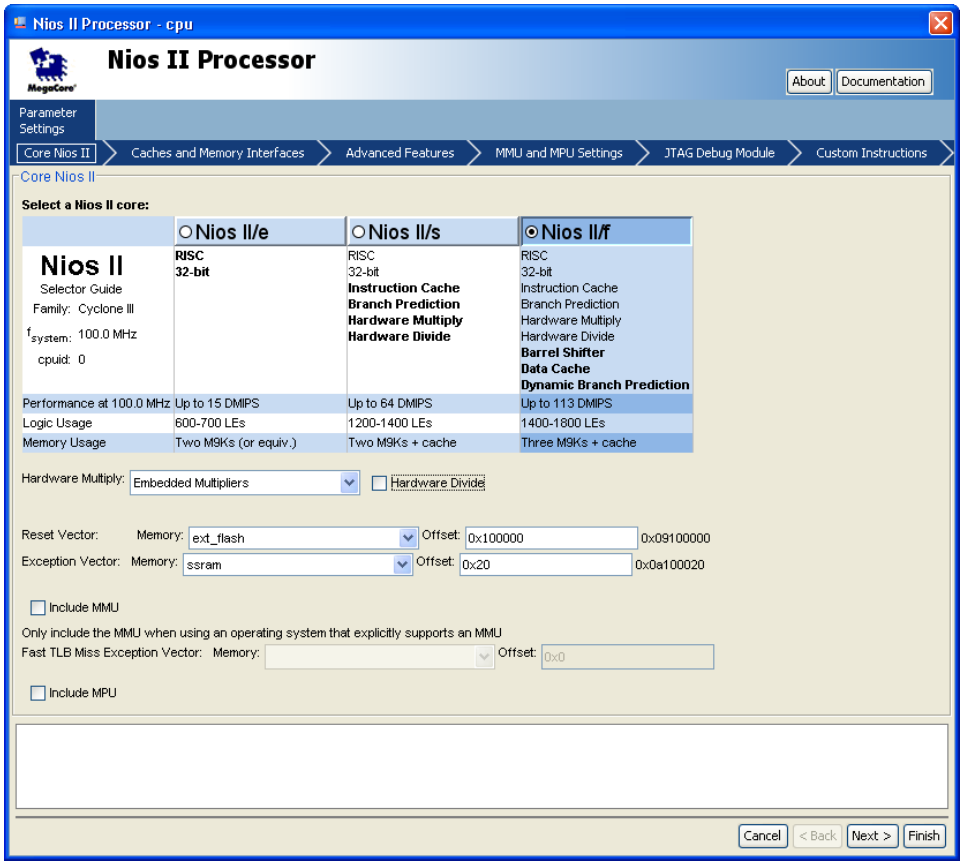
When a software exception or interrupt event occurs, the Nios II processor begins executing software in another predefined location.The exception vector specifies this location.

To set the reset and exception vectors for the Nios II processor in your system, perform the following steps:

1. In the **Module Name** column, double-click **cpu**. The Nios II processor MegaWizard interface appears.
2. Under **Reset Vector**, for **Memory**, click **ext_flash**.
3. For **Offset**, type `0x100000`.
4. Under **Exception Vector**, for **Memory**, click **ssram**.
5. For **Offset**, type `0x20`.

   Figure 3–12 shows the Nios II Processor MegaWizard interface after you perform these steps.

Figure 3–12. Nios II Processor Reset and Exception Vector Settings

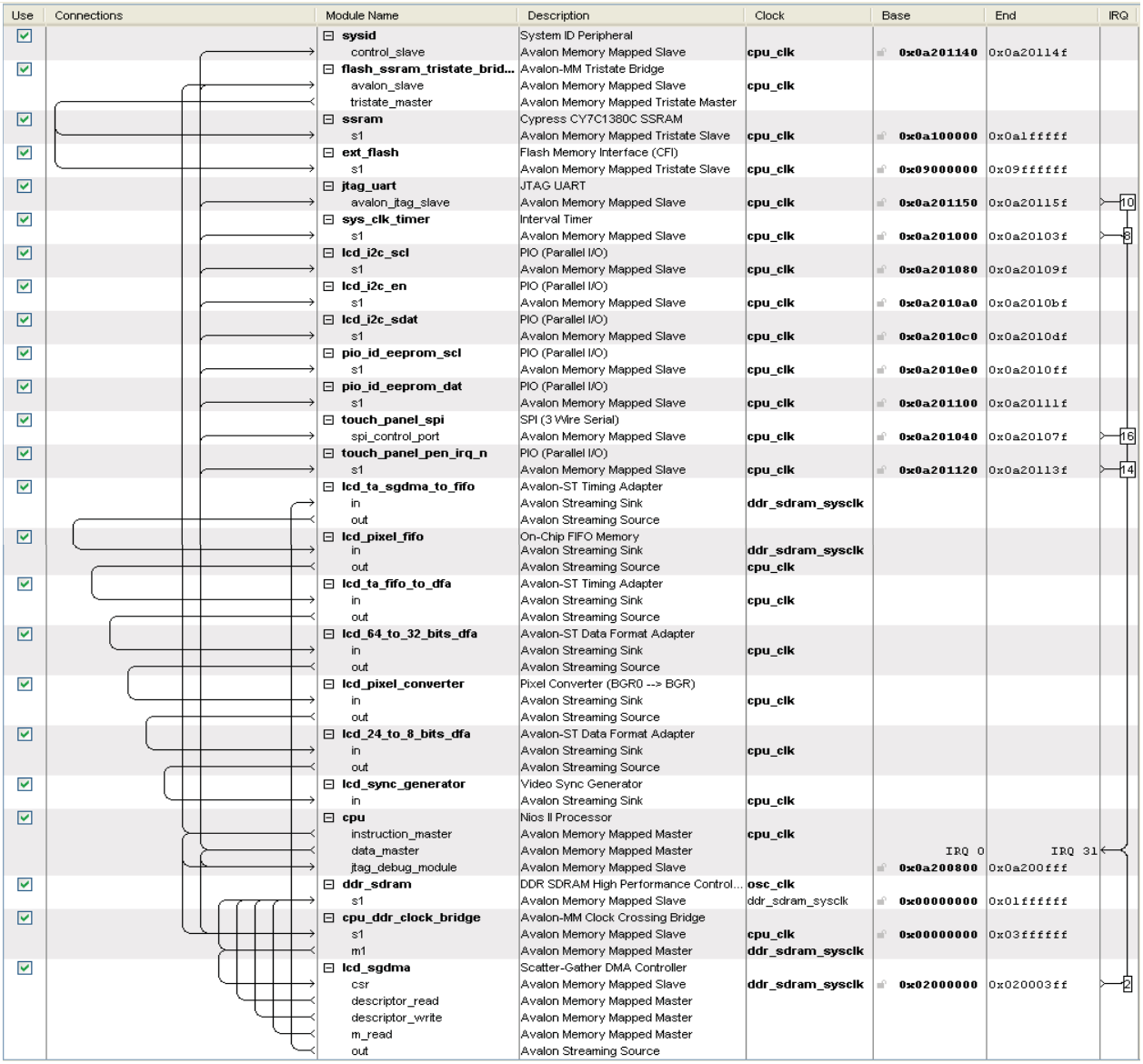


6. Click **Finish**.

Your SOPC Builder system content is complete and you can now generate the SOPC Builder system.

Figure 3–13 shows your final SOPC Builder system.

Figure 3–13. Final Digital Picture Viewer SOPC Builder System



Generate the System

After you verify that all the component names in your SOPC Builder system match the component names in Figure 3–13, you are ready to generate your system.

To generate your system, on the **System Generation** tab, click the **Generate** button.

As shown in Figure 2–1 on page 2–2, SOPC Builder creates the following output items:

- HDL files for the components in your system
- HDL files for the system interconnect to connect the components together

- System description used by the software development tool, the Nios II IDE, to build the software project

After your system generates successfully, the following message appears:

System generation was successful.

4. Completing the Quartus II Project

In this chapter you complete the Quartus II project by adding the generated SOPC Builder system to the top-level project and adding timing constraint files to your project. You compile your project in the Quartus II software to perform analysis, synthesis, fitting, place-and-route, and timing analysis. Compilation generates an FPGA image as an SRAM object file (**.sof**). After you download the FPGA image to the NEEK, the on-board FPGA functions as a processor custom-made for your application.

If you wish to work with the **.qar** provided with the tutorial files, open the **.qar** in the Quartus II software and skip directly to "Configure the FPGA" on page 4–4.

Complete the Quartus II Project

This section teaches you how to complete your Quartus II project. It contains the instructions that show you how to perform the following actions:

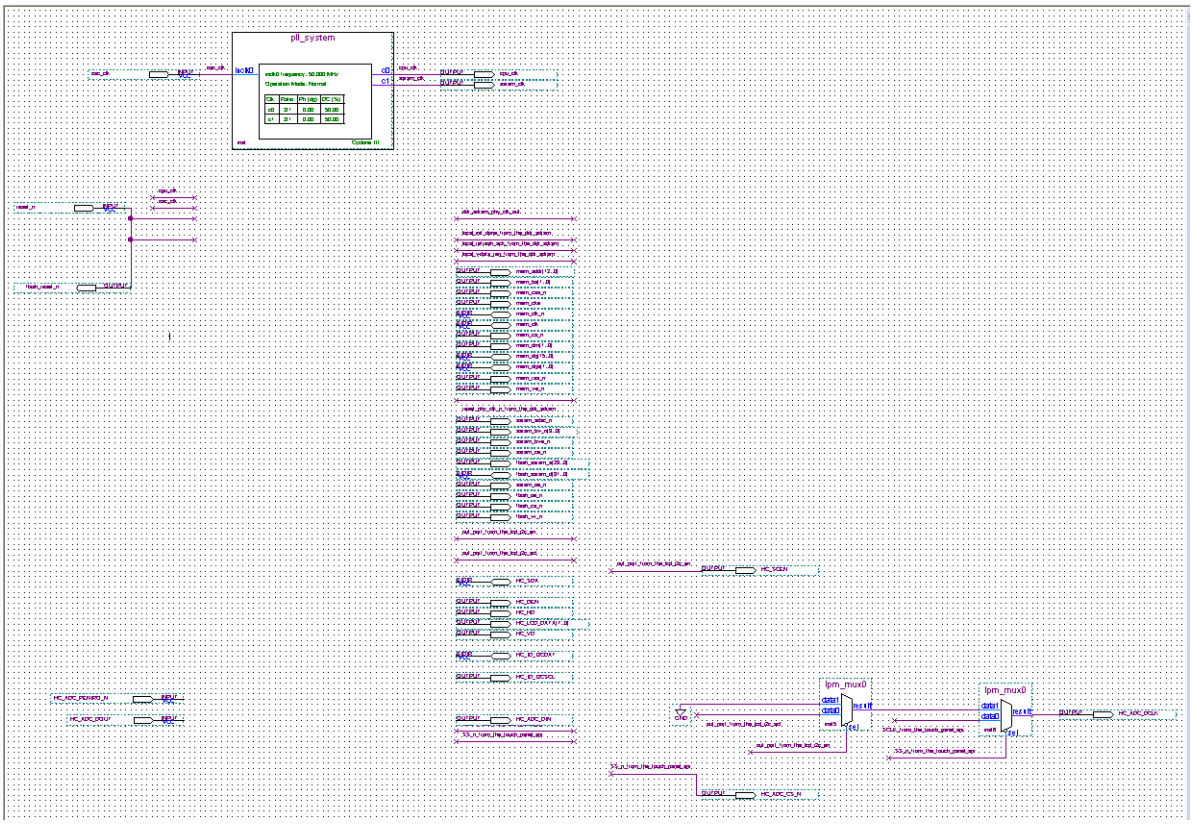
1. Open the Quartus II Project Block Diagram
2. Add the SOPC Builder System to the Quartus II Project
3. Add Timing Constraints and Compile Your Design

Open the Quartus II Project Block Diagram

To open the top-level project block diagram, perform the following steps:

1. In the Quartus II software, on the File menu, click **Open**.
2. Browse to your working directory and double-click **neek_hw_lab.bdf**. The **neek_hw_lab.bdf** Block Design File opens in the Block Editor, as shown in Figure 4–1.

Figure 4–1. NEEK Embedded Tutorial Project Top-Level Block Diagram



The Block Design file (**.bdf**) includes the pins required by the design, a PLL, a MUX, and some glue logic, but does not include the logic in the SOPC Builder system. You must add the SOPC Builder system as a new logic block to the schematic design.

Add the SOPC Builder System to the Quartus II Project

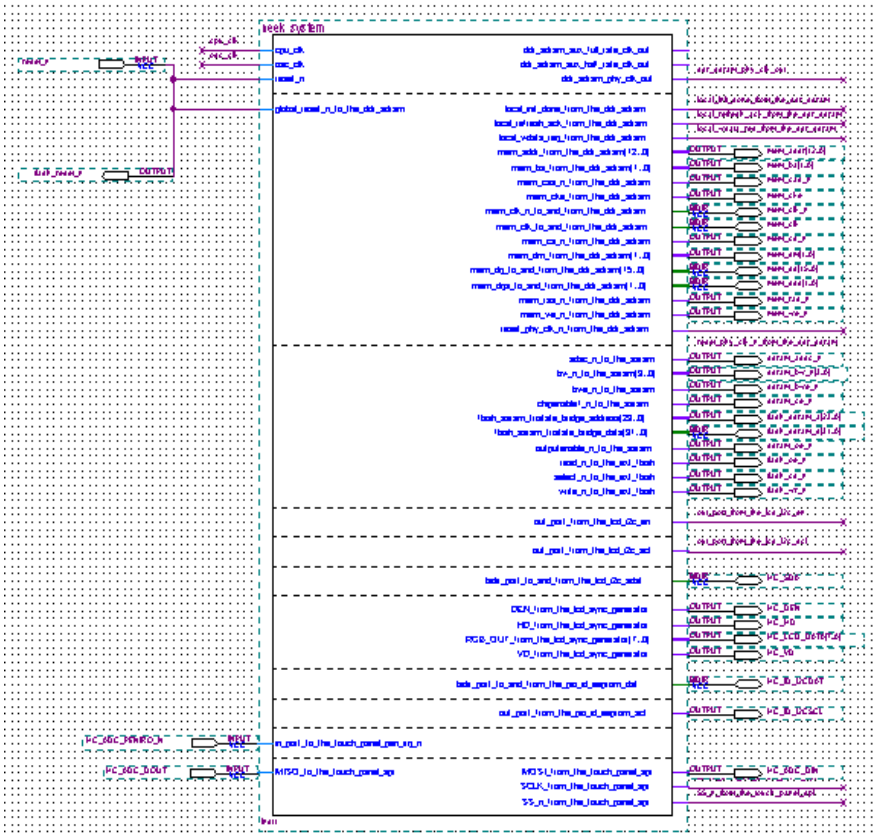
Add the SOPC Builder system to your Quartus II project by performing the following steps:

1. On the Edit Menu, click **Insert Symbol.**
2. Open the **Project** folder and highlight **neek_system**.
3. Click **OK**.
4. Place the symbol in the schematic. Align with the correct pins in the schematic, as shown in Figure 4–2.

> You may need to move some connector pins to align them with the pins of the symbol. However, if you have to move and reorder many existing configuration pins, you should reexamine your SOPC Builder system to ensure it matches Figure 3–13 on page 3–15.

Figure 4–2 shows the schematic after the SOPC Builder system is inserted.

Figure 4–2. SOPC Builder System Inserted in Top-Level Project



5. On the File menu, click **Save Project** to save the top-level schematic with the SOPC Builder system.

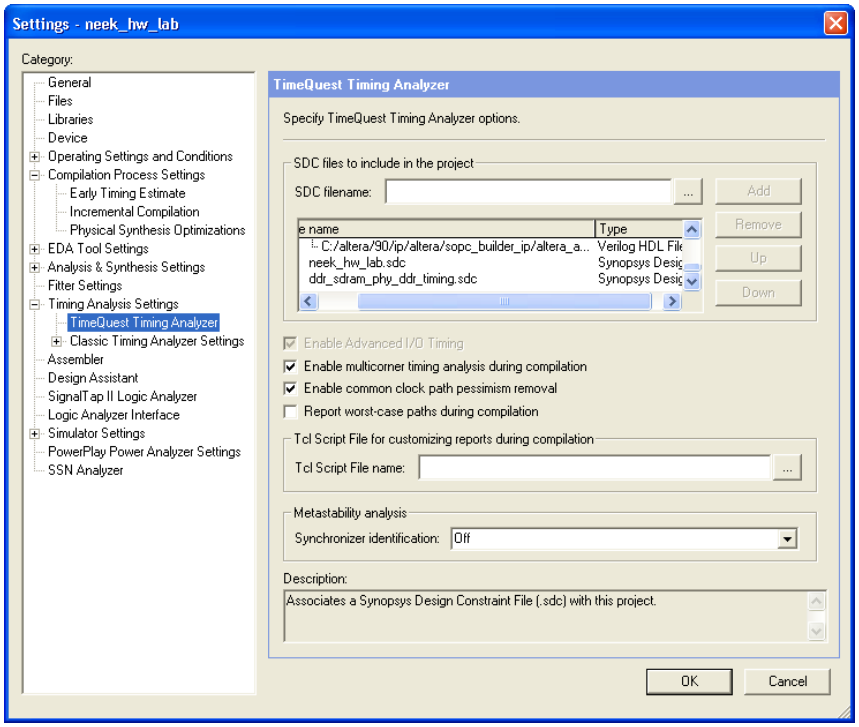
Add Timing Constraints and Compile Your Design

To ensure the design works correctly, you must add explicit timing constraints to the project. For this tutorial, the timing constraints are available in two files, **neek_hw_lab.sdc** and **ddr_sdram_phy_ddr_timing.sdc**. To add the timing constraint files to the project, perform the following steps:

1. On the Project menu, click **Add/Remove Files in Project**. The **Settings** dialog box appears.
2. Under **Category**, expand **Timing Analysis Settings** and click **TimeQuest Timing Analyzer**.
3. In the **SDC filename** box, select **neek_hw_lab.sdc.**
4. Click **Open**.
5. Click **Add**.
6. Repeat steps 3 to 5 for the file **ddr_sdram_phy_ddr_timing.sdc**.

7. Use the **Up** and **Down** buttons as necessary to ensure the sequence of the files is correct. This step is necessary to ensure the constraints are applied in the correct order. Figure 4–3 shows the required file order.

Figure 4–3. Settings Dialog Box After Two New Timing Constraint Files Added to Project



8. Click **OK**.

   The design is ready for compilation.

9. To compile your design, on the Processing menu, click **Start Compilation**.

   The Quartus II software requires a few minutes to compile the design. The design should compile without errors.

10. When compilation completes, click **OK** to close the **Full Compilation was successful** message.

Compilation produces the **.sof** file, **neek_hw_lab.sof**.

Configure the FPGA

Next, use the Quartus II Programmer to download the **.sof** to the NEEK and configure the Cyclone III device on the board.

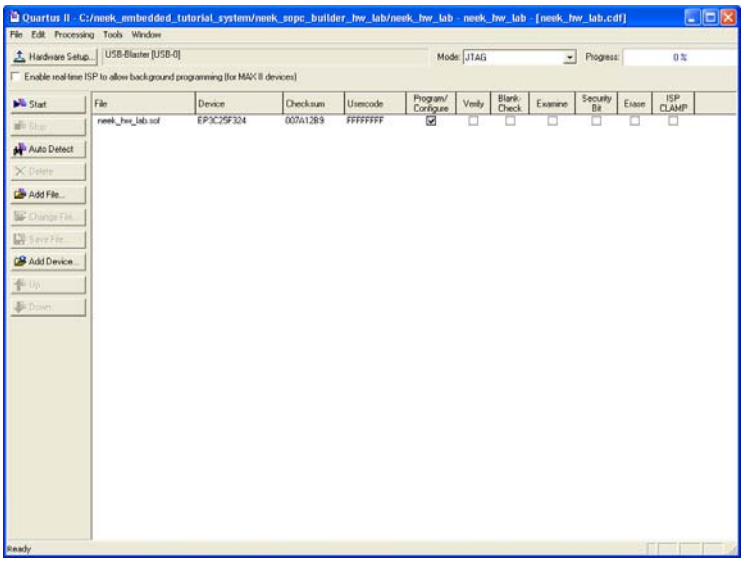
For more information about the Quartus II Programmer, refer to the *Quartus II Programmer* chapter in volume 3 of the *Quartus II Handbook*.

To configure the FPGA, perform the following steps:

1. On the Tools menu, click **Programmer**.

2. Ensure that the power switch (SW1) on the back side of the NEEK is in the OFF position (up).
3. Connect the USB cable from your PC to the NEEK.
4. Connect the power cable to the NEEK.
5. Press the power switch to turn on the NEEK.
6. In the Quartus II Programmer, click **Hardware Setup**.
7. Under **Currently selected hardware**, click **USB-Blaster [USB-0]**.
8. Click **Close.**
9. Click **Auto Detect**. The **Device** column lists the EP3C25F324 device on your board, indicating that the Programmer detected the device. Figure 4–4 shows the Quartus II Programmer after you perform this step.

Figure 4–4. Quartus II Programmer



10. If your **.sof** does not appear in the Programmer, perform the following steps:
    a. In the **File** column, click **<none>**.
    b. Click the **Change File** button.
    c. Browse to your working directory.
    d. Select **neek_hw_lab.sof**.
    e. Click **Open**.
11. Turn on **Program/Configure** for your **.sof**.
12. Click **Start** to program the FPGA.

After the Programmer completes programming the FPGA, the progress bar reaches **100%**, and no error messages appear.

After you complete all the steps in this chapter, you have successfully compiled and downloaded the FPGA image to the Cyclone III device on your NEEK. The processor is ready to run. Next, you must develop the software application and download it to the NEEK.

5. Creating the Software Application

In this chapter you use the Nios II IDE to develop the software application that runs on your system. You create a new software application project, add the software source files to the project, configure the project, and build it. The result of the build process is an executable and linkable file (**.elf**). The Nios II IDE downloads the application **.elf** to the memory location at which the Nios II processor expects to find the executable image. The Nios II processor can then run the application **.elf**.

For more information about the Nios II IDE, refer to the *Nios II Integrated Development Environment* chapter of the *Nios II Software Developer's Handbook*, to the Nios II IDE online Help, and to the *Developing Nios II Software* chapter of the *Embedded Design Handbook*.

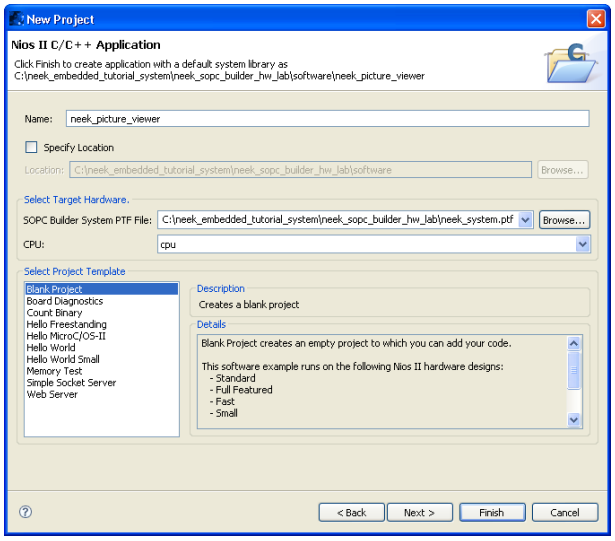
Create a New Software Project in the Nios II IDE

To create your software project in the Nios II IDE, perform the following steps:

1. To start the Nios II IDE, in SOPC Builder, on the **System Generation** tab, click **Nios II IDE**.
2. On the File menu, under **New**, click **Project**. The **New Project** dialog box appears.
3. In the **New Project** dialog box, click **Nios II C/C++ Application**.
4. Click **Next**.
5. Under **Name**, type `neek_picture_viewer`.
6. To set the **SOPC Builder System PTF File**, click the **Browse** button to locate the **neek_system.ptf** file in your working directory
7. Under **Select Project Template**, click **Blank Project**.

Figure 5–1 shows the **New Project** dialog box after you perform these steps.

Figure 5–1. New Project Wizard Settings for Tutorial Software Application Project



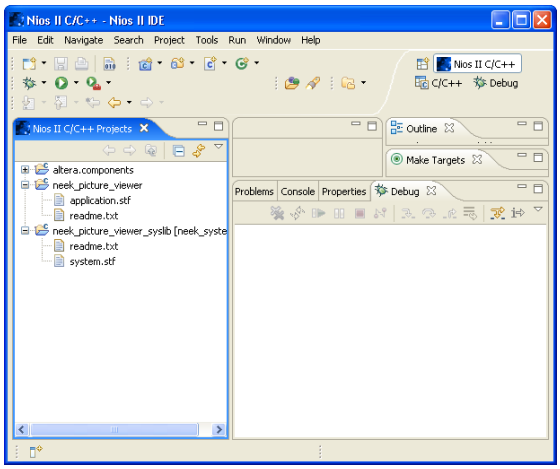
8. Click **Next**.
9. Turn on **Create a new system library.**
10. Click **Finish.**
11. If you are prompted to open the Nios II C/C++ perspective, click **Yes**.

The Nios II IDE creates the following two new software project directories:

- **neek_picture_viewer**—The software application project directory.
- **neek_picture_viewer_syslib**—The system library project for the software application project. The system library project builds the system library drivers for the components in your SOPC Builder system.

Figure 5–2 shows the new software project directories in the Nios II IDE.

Figure 5–2. New Project Directories in Nios II IDE



Because this software project is based on the blank template, the project directories do not yet contain any source code files.

Add Source Code Files to the Project

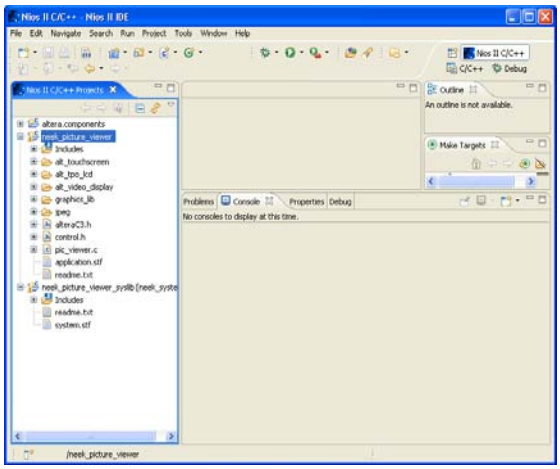
Your working directory contains a **software_source_files** folder. The files in this folder are the source code files for the tutorial software application project.

To copy these files to the software application folder, perform the following steps:

1. In Windows Explorer, open the **software_source_files** folder and select all the files.
2. Drag the files and drop them on the **neek_picture_viewer** folder in the Nios II IDE.

   The files are copied to the software project directory, and their names appear in the Nios II IDE as shown in Figure 5–3.

Figure 5–3. Software Application Project with Source Code Files



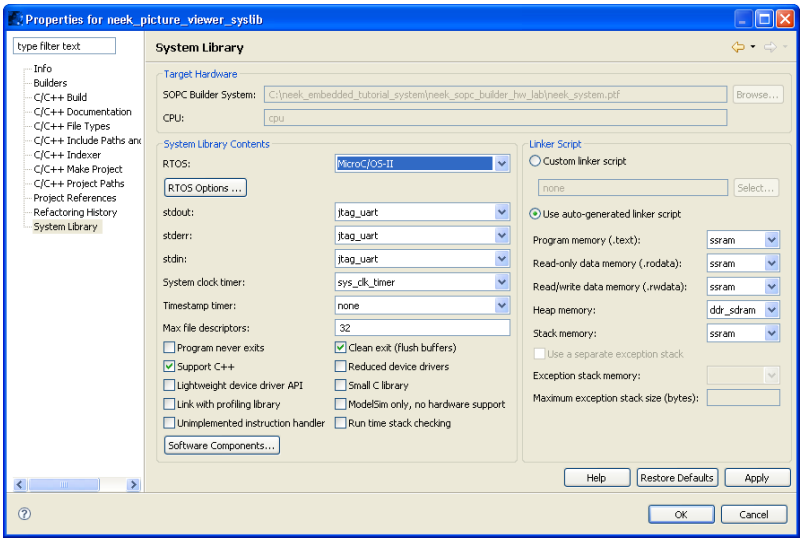
Configure System Library Build Properties

In this section, you configure the system library build to specify the properties of this software system. These properties include the interface for stdio and stderr messages, the memory in which to allocate the stack and the heap, and whether an operating system or network stack should be included in the project image.

To configure the system library build properties, perform the following steps:

1. On the Nios II IDE **Nios II C/C++ Projects** tab, right-click **neek_picture_viewer_syslib**.
2. Click **Properties**.
3. In the left pane, click **System Library**. The **System Library** dialog box appears.
4. Under **RTOS**, click **MicroC/OS-II**. Figure 5–4 shows the **System Library** settings.

Figure 5–4. System Library Settings for Digital Picture Viewer



The software design uses the MicroC/OS-II operating system. All stdout, stdin, and stderr messages are directed to the **jtag_uart**. You use the auto-generated linker script to store the heap linker section in DDR SDRAM memory and the other linker sections—program memory, read-only data memory, read/write data memory, and stack—in SSRAM memory.

5. Click **Apply**.
6. Click **OK**.

Specify the Zip File System Settings

The tutorial software application project manipulates stored JPEG images. These JPEG images are stored in the CFI flash memory on the NEEK. The project uses the Altera read-only zip file system to store and access the JPEG images in flash memory.

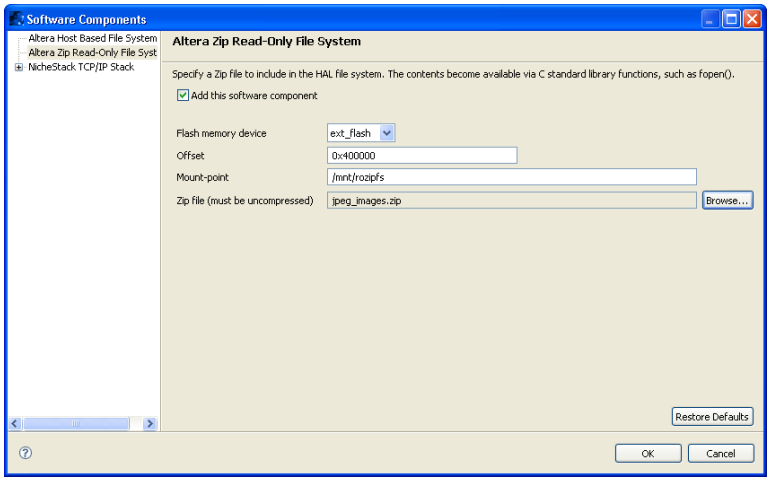
To use the read-only zip file system, you must ensure that the JPEG image files are zipped in a single file in uncompressed mode. The tutorial design files include a prezipped, uncompressed file that contains the JPEG image files.

To add the Altera read-only zip file system and set up your project to store the prezipped JPEG images to the NEEK flash memory, perform the following steps:

1. In Windows Explorer, open the **images** folder in your working directory. The list of files in this folder includes the **jpeg_images.zip** file.
2. To copy the images to your project, drag and drop **jpeg_images.zip** on the **neek_picture_viewer_syslib** folder in the Nios II IDE.
3. Right-click **neek_picture_viewer_syslib**.
4. Click **Properties**.
5. In the left pane, click **System Library**. The **System Library** dialog box appears.
6. Click **Software Components**. The **Software Components** dialog box appears.

7. Click **Altera Zip Read-Only File System**.
8. Turn on **Add this software component**.
9. Under **Offset**, type `0x400000`.
10. Click **Browse**.
11. Click **jpeg_images.zip**.
12. Click **OK**.

Figure 5–5. Altera Read-Only Zip File System Settings for Digital Picture Viewer



13. Click **OK**.
14. Click **OK**.

For more information about the Altera read-only zip file system, refer to the *Read-Only Zip File System* chapter of the *Nios II Software Developer's Handbook*.

Build the Software Project

To compile and build the image files for your software project, perform the following steps:

1. Right-click **neek_picture_viewer**.
2. Click **Build Project**.

The Nios II IDE compiles the associated system library, and then the software project. The result of the compilation process is an executable and linkable format file (**.elf**) for the application and a Motorola S-record file (**.flash**) that contains the JPEG images for the zip file system.

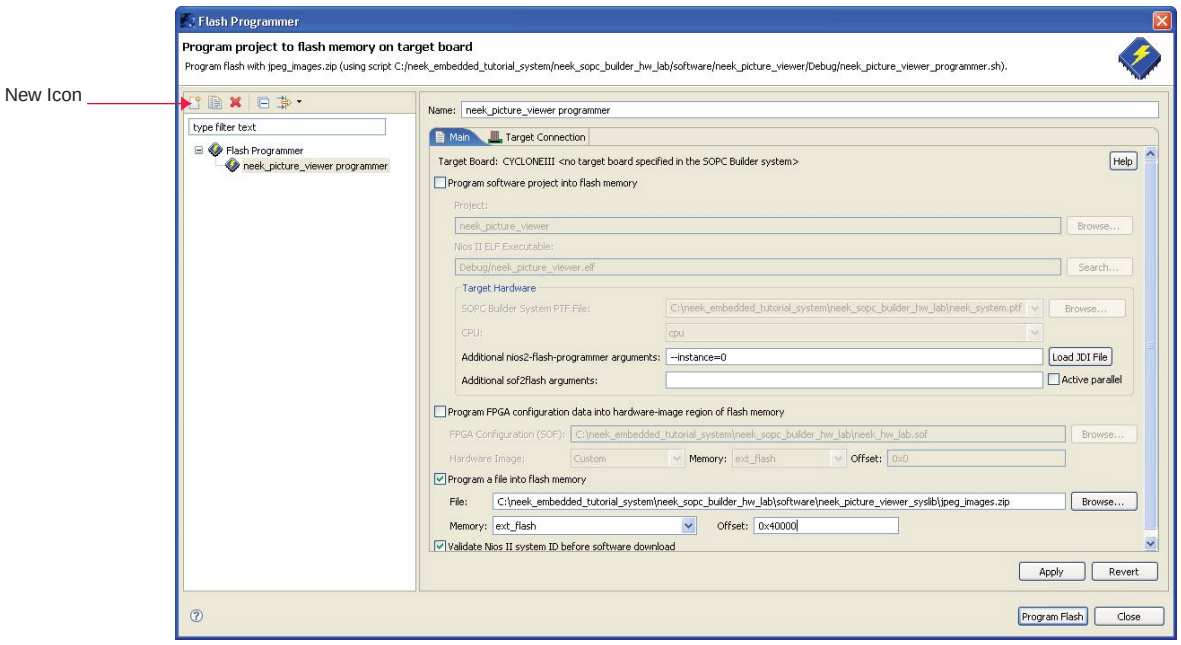
Program the Zip File to Flash Memory

You use the Flash Programmer to program the JPEG images in the zip file system in the flash memory on the NEEK.

To program the flash memory with the **.flash** you created in the build process, perform the following steps:

1. In the Nios II IDE, on the Tools menu, click **Flash Programmer**.
2. Click the **New** icon to create a new configuration. Figure 5–6 shows the **New** icon.

Figure 5–6. Flash Programmer Settings to Program Flash Memory With JPEG Images .zip File



3. Turn on **Program a file into flash memory**.
4. In the **File** box, browse to and select the **jpeg_images.zip** file in the **neek_picture_viewer_syslib** folder, as shown in Figure 5–6.
5. In the **Memory** box, select **ext_flash**.
6. In the **Offset** box, type `0x400000`.
7. Turn on **Validate Nios II system ID before software download**.
8. On the **Target Connection** tab, ensure **JTAG cable** is set to **USB-Blaster**.
9. Click **Apply**.
10. Click **Program Flash**.
11. If you are prompted to confirm you want to program the flash memory now, click **Yes**.

For more information about the Flash Programmer, refer to the *Nios II Flash Programmer User Guide*.

Run the Software Application on the NEEK

Running an application on the target hardware requires the following two image files:

- FPGA hardware image SRAM Object File (**.sof**)
- Software executable image (**.elf**)

In Chapter 4, Completing the Quartus II Project, you configured the FPGA by downloading the .**sof** to the FPGA. After you configure the FPGA, it is ready to run the software application. You download the application through the USB cable that remains plugged in to the NEEK board.

To run the software project on the Nios II processor you configured on the NEEK, perform the following steps:

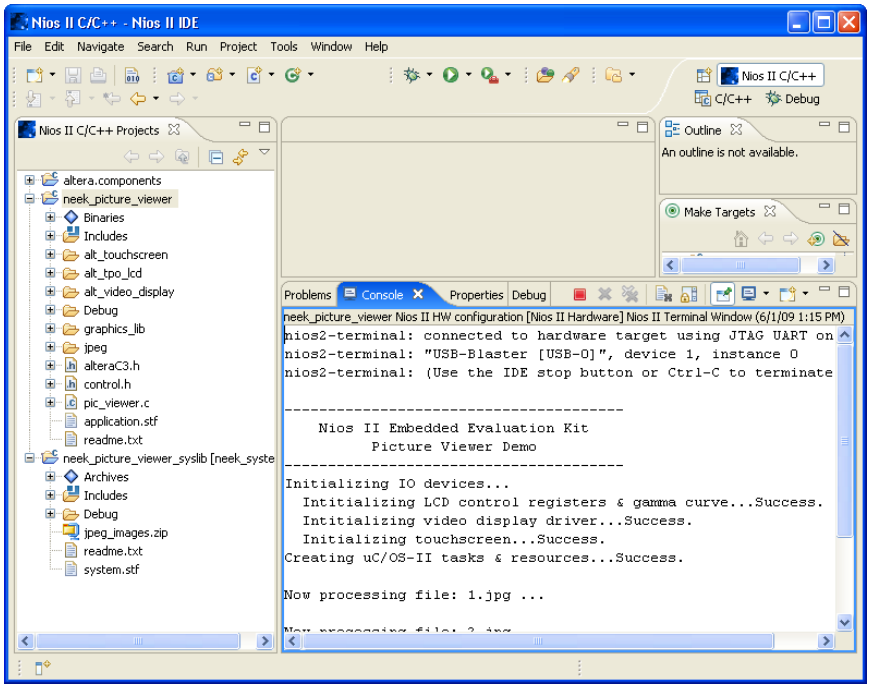
1. Right-click **neek_picture_viewer**.
2. Click **Run As** > **Nios II Hardware**.
3. If the message **You have more than one JTAG cable available** appears in the Nios II IDE Console window, perform the following steps:
   a. On the Run **menu**, click **Run.**
   b. On the **Target Connection** tab, for **JTAG cable**, click **USB-Blaster**.
   c. Click the **Run** button.

The **Run** or **Run As** command rebuilds the software project to create an up–to-date executable and then download the code to memory. The debugger resets the Nios II processor, and the Nios II processor then executes the downloaded code.

Console Output

After the application begins executing, it relays messages to the Nios II IDE through the JTAG interface. Figure 5–7 shows the output in the Nios II IDE Console window.

Figure 5–7. Output from Successful Run of Digital Picture Viewer



Interact with the Software Application

When the neek_picture_viewer application starts running on the Nios II processor, the digital picture viewer displays the first JPEG image. To display the next image, press the **Forward** button on the LCD touch screen. To display the previous image, press the **Back** button on the LCD touch screen. Two small thumbnails on the bottom of the screen display the decoding status of the previous and next images.

Congratulations! You have successfully completed the entire embedded systems tutorial.

The Flash Programmer overwrites the default Application Selector on the NEEK. To restore the factory image, refer to the "Restoring the Factory Image" appendix in the *Nios II Embedded Evaluation Kit, Cyclone III Edition User Guide*.

Taking the Next Step

Altera offers many resources to help you get started designing your own embedded designs. If you are starting from scratch, Altera recommends that you purchase a Nios II development kit. Each kit includes pre-generated Nios II processor systems and a Nios II IP license.

If you already have a working project, you can add an SOPC Builder system to your top-level project as a stub or add your design to the SOPC Builder system as a custom component.

The following documents and web pages provide information to help you create custom components for your SOPC Builder system:

- *SOPC Builder Component Development Walkthrough* chapter in Volume 4: SOPC Builder of the *Quartus II Handbook* shows you how to create a custom SOPC Builder component.
- *Component Editor* chapter in Volume 4: SOPC Builder of the *Quartus II Handbook* describes how to add a custom component to an SOPC Builder system using the component editor.
- Avalon Memory-Mapped Master Templates web page describes how to use an Avalon-MM template to create an Avalon-MM custom component.
- Avalon Memory-Mapped Slave Template web page describes how to use an Avalon-MM template to create an Avalon-MM slave custom component.
- *AN459: Guidelines for Developing a Nios II HAL Device Driver* describes how to develop software device drivers for your custom component.

The following Altera documents and web pages provide background and general information about building embedded systems using Altera tools:

Subject	Information Source
Altera embedded solutions	www.altera.com/embedded
Embedded IP peripherals	*Volume 5: Embedded Peripherals* of the *Quartus II Handbook*
Embedded system design	*Embedded Design Handbook*
Evaluation and development kits	www.altera.com/devkits
Nios II processor	*Nios II Processor Reference Handbook*
Nios II processor system design examples and software applications	Nios II Embedded Processor Design Examples web page
Nios II software development tools	*Nios II Software Developer's Handbook*
Nios II user community	Nios II user community forum: www.niosforum.com Nios II user community wiki: www.nioswiki.com

Subject	Information Source
Online tutorials and in-person training	Embedded Hardware Designer Training Embedded Software Designer Training
SOPC Builder	*Volume 4: SOPC Builder* of the *Quartus II Handbook*

How to Contact Altera

For the most up-to-date information about Altera products, refer to the following table.

Contact *(Note 1)*	Contact Method	Address
Technical support	Website	www.altera.com/support
Technical training	Website	www.altera.com/training
	Email	custrain@altera.com
Product literature	Website	www.altera.com/literature
Non-technical support (General)	Email	nacomp@altera.com
(Software Licensing)	Email	authorization@altera.com

Note:

(1) You can also contact your local Altera sales office or sales representative.

Revision History

The following table displays the revision history for this document.

Date & Document Version	Changes Made	Summary of Changes
June 2009 v1.0	Initial release.	—

Typographic Conventions

The following table shows the typographic conventions that this document uses.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Indicates command names, dialog box titles, dialog box options, and other GUI labels. For example, **Save As** dialog box.
bold type	Indicates directory names, project names, disk drive names, file names, file name extensions, and software utility names. For example, **\qdesigns** directory, **d:** drive, and **chiptrip.gdf** file.
Italic Type with Initial Capital Letters	Indicates document titles. For example, *AN 519: Stratix IV Design Guidelines.*
Italic type	Indicates variables. For example, *n* + 1. Variable names are enclosed in angle brackets (< >). For example, *<file name>* and *<project name>***.pof** file.
Initial Capital Letters	Indicates keyboard keys and menu names. For example, Delete key and Options menu.

© June 2009 Altera Corporation

Visual Cue	Meaning
“Subheading Title”	Quotation marks indicate references to sections within a document and titles of Quartus II Help topics. For example, “Typographic Conventions”.
`Courier type`	Indicates signal, port, register, bit, block, and primitive names. For example, `data1`, `tdi`, and `input`. Active-low signals are denoted by suffix `n`. For example, `resetn`. Indicates command line commands and anything that must be typed exactly as it appears. For example, `c:\qdesigns\tutorial\chiptrip.gdf`. Also indicates sections of an actual file, such as a Quartus II report file, references to parts of files (for example, the AHDL keyword `SUBDESIGN`), and logic function names (for example, `TRI`).
1., 2., 3., and a., b., c., and so on.	Numbered steps indicate a list of items when the sequence of the items is important, such as the steps listed in a procedure.
■ ■	Bullets indicate a list of items when the sequence of the items is not important.
	The hand points to information that requires special attention.
	A caution calls attention to a condition or possible situation that can damage or destroy the product or your work.
	A warning calls attention to a condition or possible situation that can cause you injury.
	The angled arrow instructs you to press the Enter key.
	The feet direct you to more information about a particular topic.